Interrupt

Arduino, STM32, Raspberry Pi

INTERRUZIONE

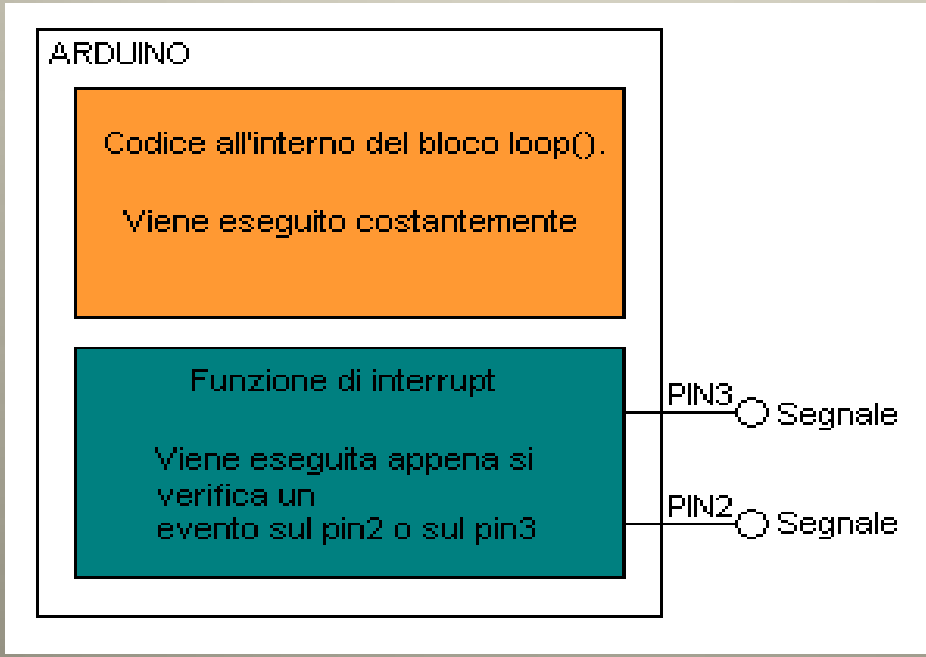
- Normalmente il micro esegue delle istruzioni in modo sequenziale e ripetitivo ma che possono essere interrotte in uno dei seguenti modi:
  - Si interrompe l'alimentazione
  - Viene interrotto il flusso delle istruzioni all'interno del loop() ed invocate altre routine. In questo caso si parla di interrupt e, quando le routine terminano, il flusso del programma prosegue normalmente.

DEFINIZIONE di Interrupt

- Segnale asincrono inviato da una periferica collegata al microcontrollore.
- Viene generato quando si verifica una variazione di stato su un pin del micro.
- Viene gestito dal microcontrollore ed è controllabile via software tramite delle apposite istruzioni
- È chiaro che l'interrupt è molto più utile che una semplice interruzione di alimentazione perché possiamo avere la necessità di gestire in background alcune routine ed eseguire istantaneamente un'operazione nel caso si manifesti un evento asincrono esterno.
- Quando viene attivata una interruzione quindi, è possibile eseguire del codice in modo automatico, interrompendo momentaneamente il normale flusso di codice
- Bisogna ricordare che l'interrupt è un evento che viene eseguito con la massima priorità

Interrupt di Arduino uno

- Per Arduino Uno, la funzione interrupt si applica su due0 pin dedicati: il pin 2 e il pin 3



Interrupt Arduino

- Per gestire l'interrupt di Arduino c'è bisogno di un codice
- Nelle variabili globali viene dichiarata la seguente funzione:
  - attachInterrupt(0/1, funzione, LOW/CHANGE/FALLING/RISING);
- Se si seleziona 0, si attende l'interrupt sul pin 2; se si seleziona 1, si attende l'interrupt sul pin 3
- Si può scegliere di attivare l'interruzione momentanea del loop n uno dei seguenti casi:
  - LOW il segnale è a livello logico basso
  - CHANGE il segnale sta cambiando di stato
  - FALLING il segnale cambia stato da livello logico alto a quello basso
  - RISING il segnale cambia stato da livello logico basso a quello alto

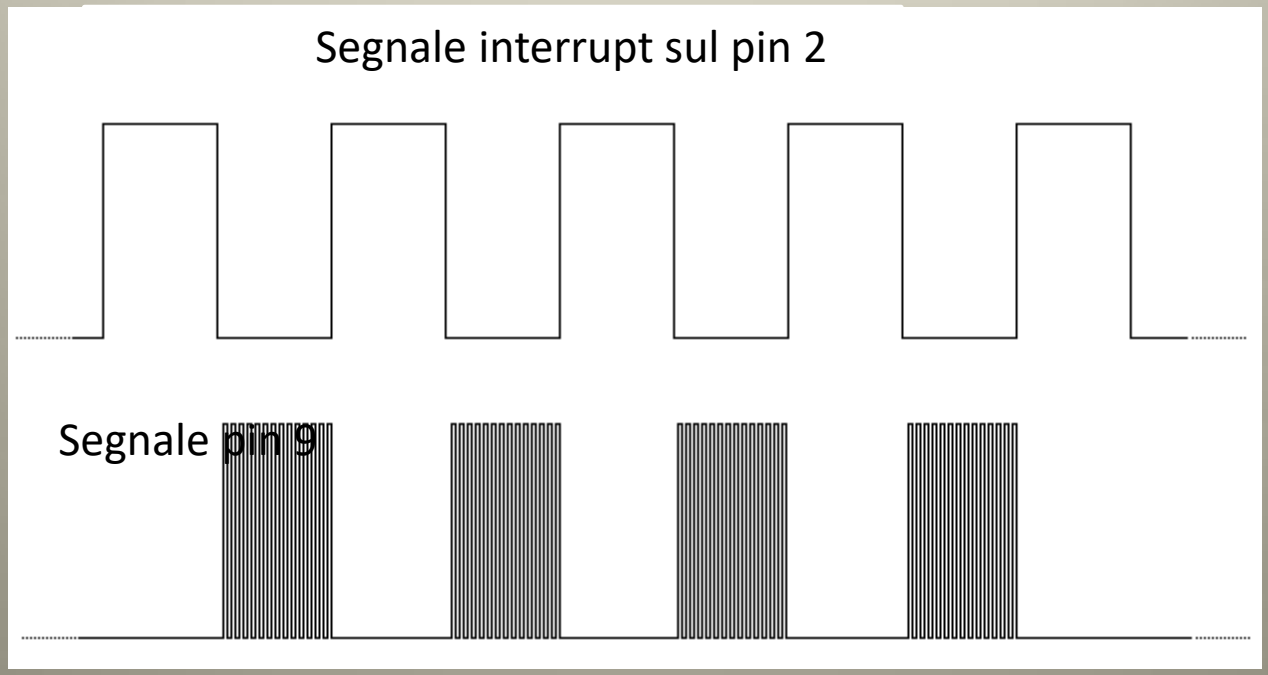
Interrupt Arduino: esempio LOW

- Supponiamo di voler cambiare lo stato di un led sul pin 9 ogni volta che si genera un interrupt portando a livello basso il pin 2

```
int led = 9;
    volatile int state = LOW;
void setup()
    {
    pinMode(led, OUTPUT);
    digitalWrite(led, LOW);
    attachInterrupt(0, gestione, LOW);
    }
void loop()
    {
    delay(10);
    }
void gestione()
    {
    state = !state;
    digitalWrite(led, state);
    }
```

Parametro LOW

- Ogni volta che il segnale sul pin 2 è basso, si genera un interrupt che cambia lo stato del led



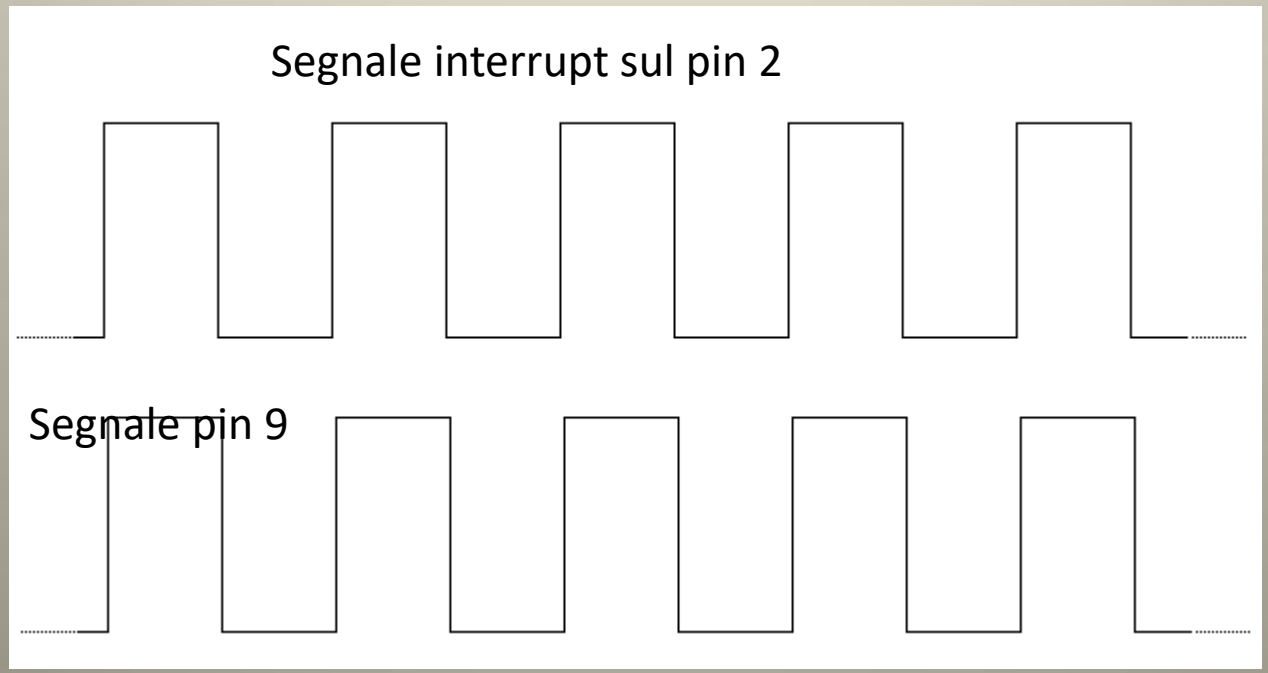
Esempio CHANGE

- Supponiamo di voler cambiare lo stato di un led sul pin 9 ogni volta che si genera un interrupt cambiando lo stato del pin 2

```
int led = 9;
    volatile int state = LOW;
void setup()
    {
    pinMode(led, OUTPUT);
    digitalWrite(led, LOW);
    attachInterrupt(0, gestione, GHANGE);
    }
void loop()
    {
    delay(10);
    }
void gestione()
    {
    state = !state;
    digitalWrite(led, state);
    }
```

Parametro GHANGE

Ogni volta che il segnale sul pin 2 cambia, si genera un interrupt che cambia lo stato del led



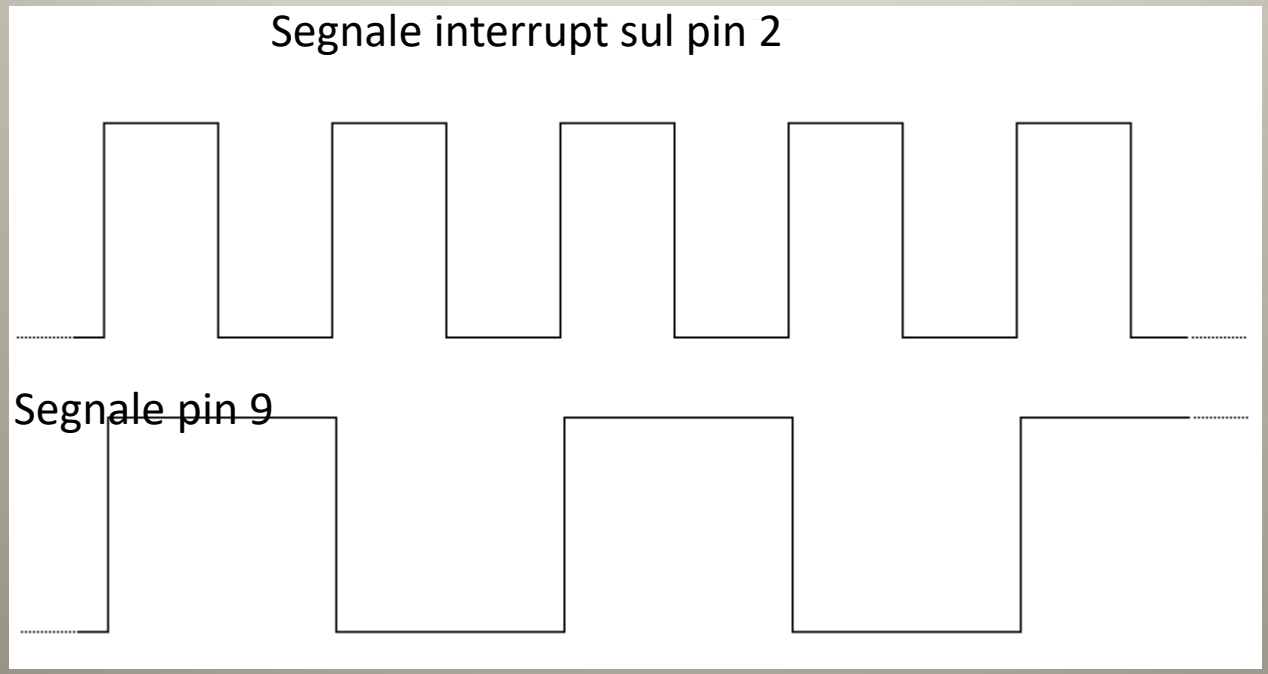
Esempio RISING

- Supponiamo di voler cambiare lo stato di un led sul pin 9 ogni volta che si genera un interrupt quando il livello logico del pin 2 passa da basso ad alto

```
int led = 9;
    volatile int state = LOW;
void setup()
    {
    pinMode(led, OUTPUT);
    digitalWrite(led, LOW);
    attachInterrupt(0, gestione, RISING);
    }
void loop()
    {
    delay(10);
    }
void gestione()
    {
    state = !state;
    digitalWrite(led, state);
    }
```

Parametro RISING

Ogni volta che il segnale sul pin 2 cambia da basso ad alto, si genera un interrupt che cambia lo stato del led



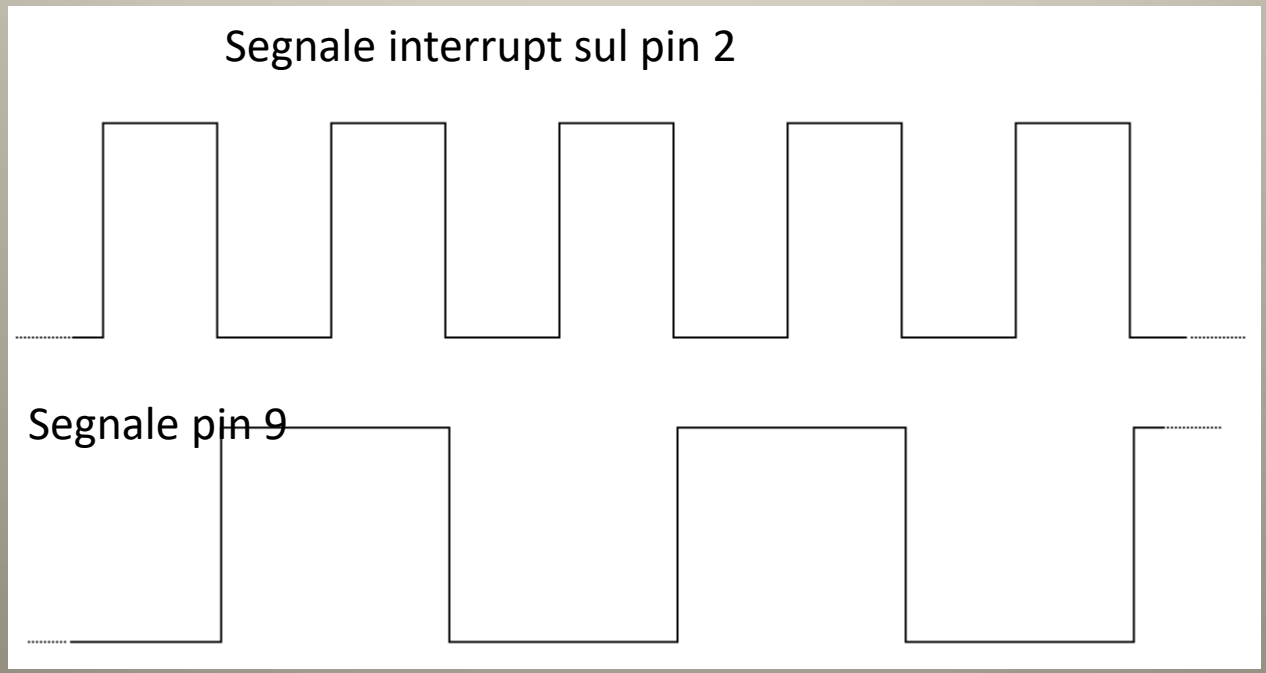
Esempio FALLING

- Supponiamo di voler cambiare lo stato di un led sul pin 9 ogni volta che si genera un interrupt quando il livello logico del pin 2 passa da alto a basso

```
int led = 9;
    volatile int state = LOW;
void setup()
    {
    pinMode(led, OUTPUT);
    digitalWrite(led, LOW);
    attachInterrupt(0, gestione, FALLING);
    }
void loop()
    {
    delay(10);
    }
void gestione()
    {
    state = !state;
    digitalWrite(led, state);
    }
```

Parametro FALLING

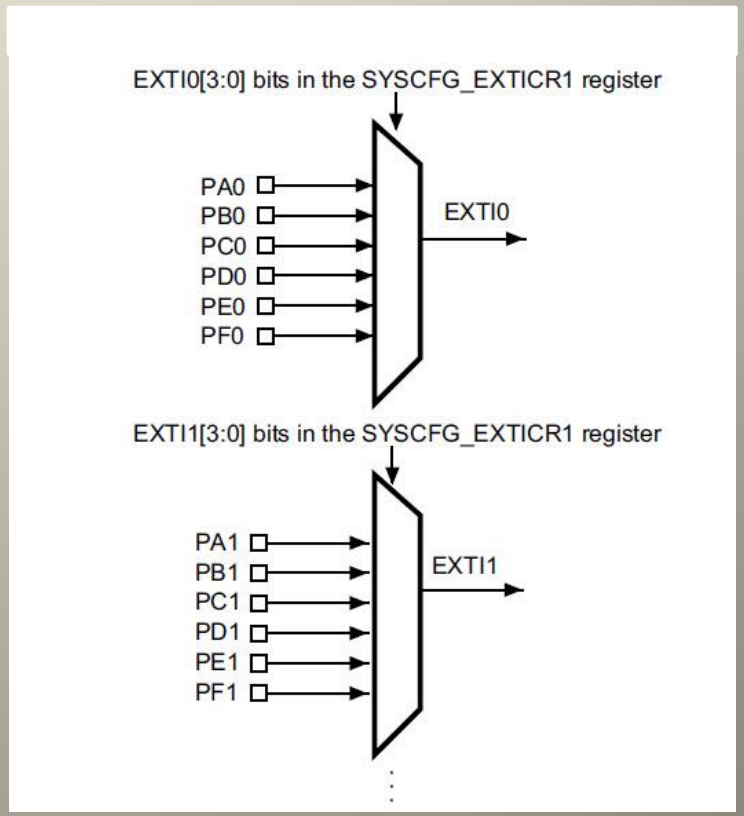
Ogni volta che il segnale sul pin 2 cambia da alto a basso, si genera un interrupt che cambia lo stato del led



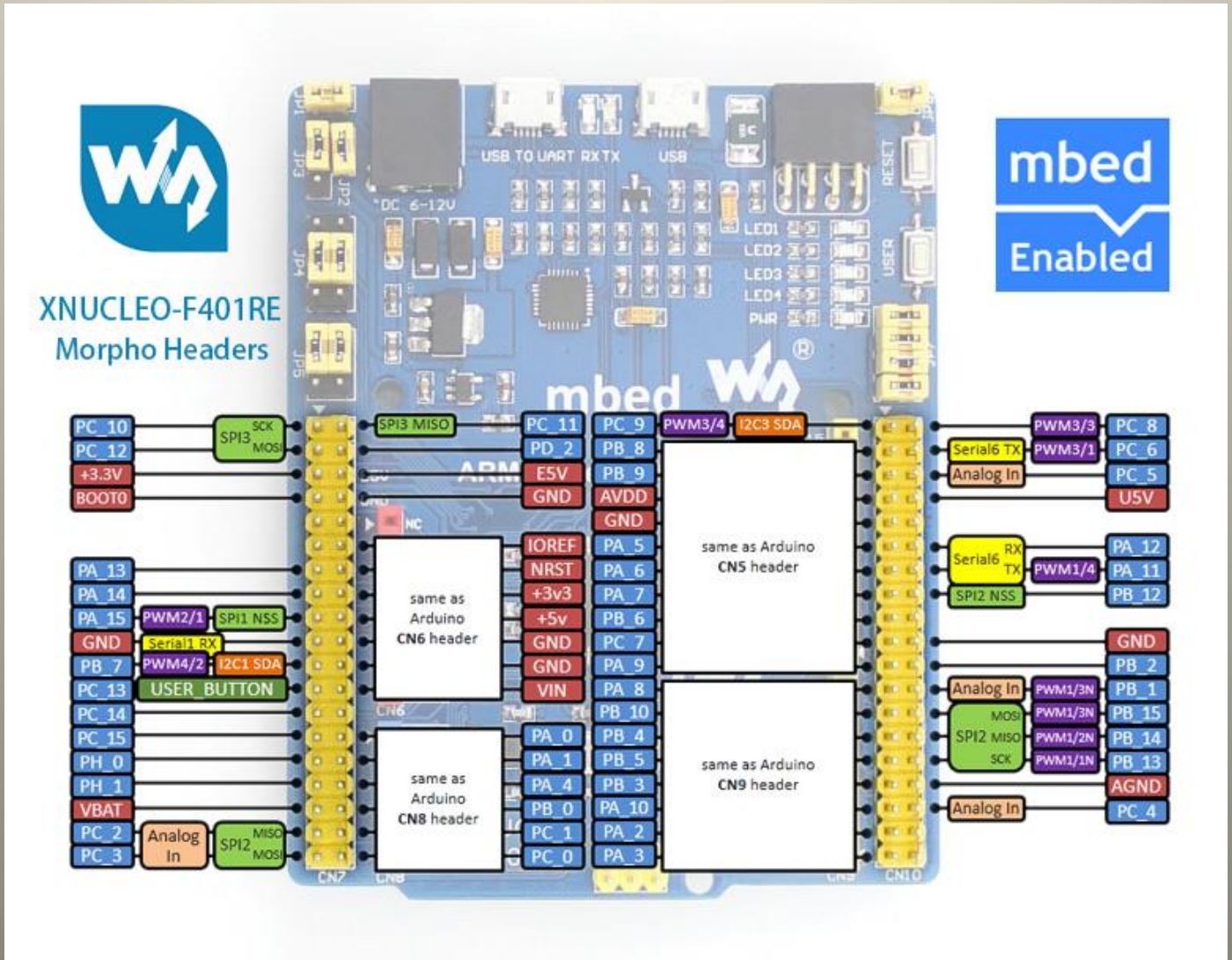
STM32

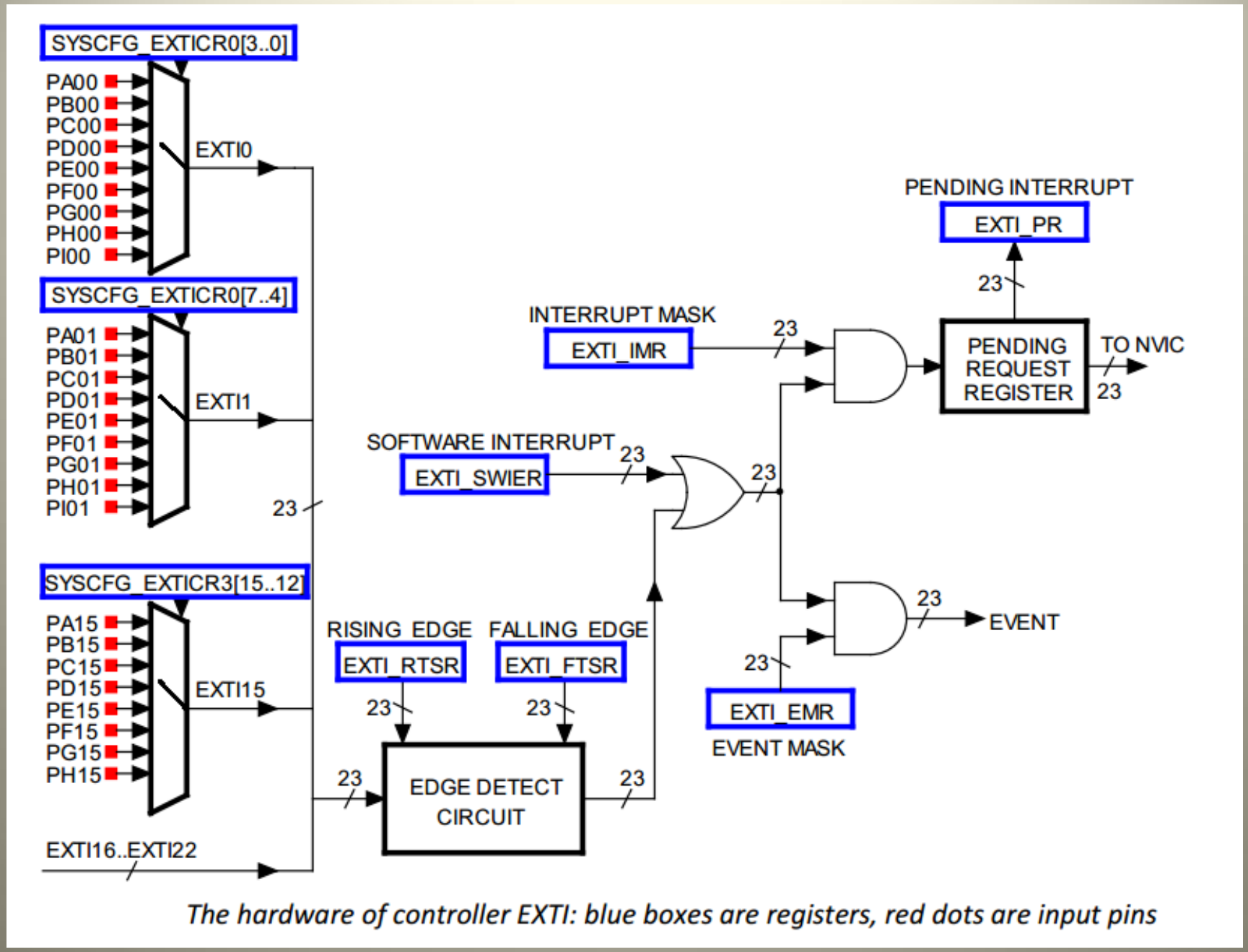
- Tutti i pin possono essere utilizzati per interrupt
- Gli interrupt possono essere utilizzati contemporaneamente ma con una regola
- I pin interrupt sono multiplexati in questo modo:

I pin PA0 e PB0 non possono essere attivati contemporaneamente come interrupt
I pin PA0 e PB1 possono essere attivati Contemporaneamente come interrupt



Pin stm32

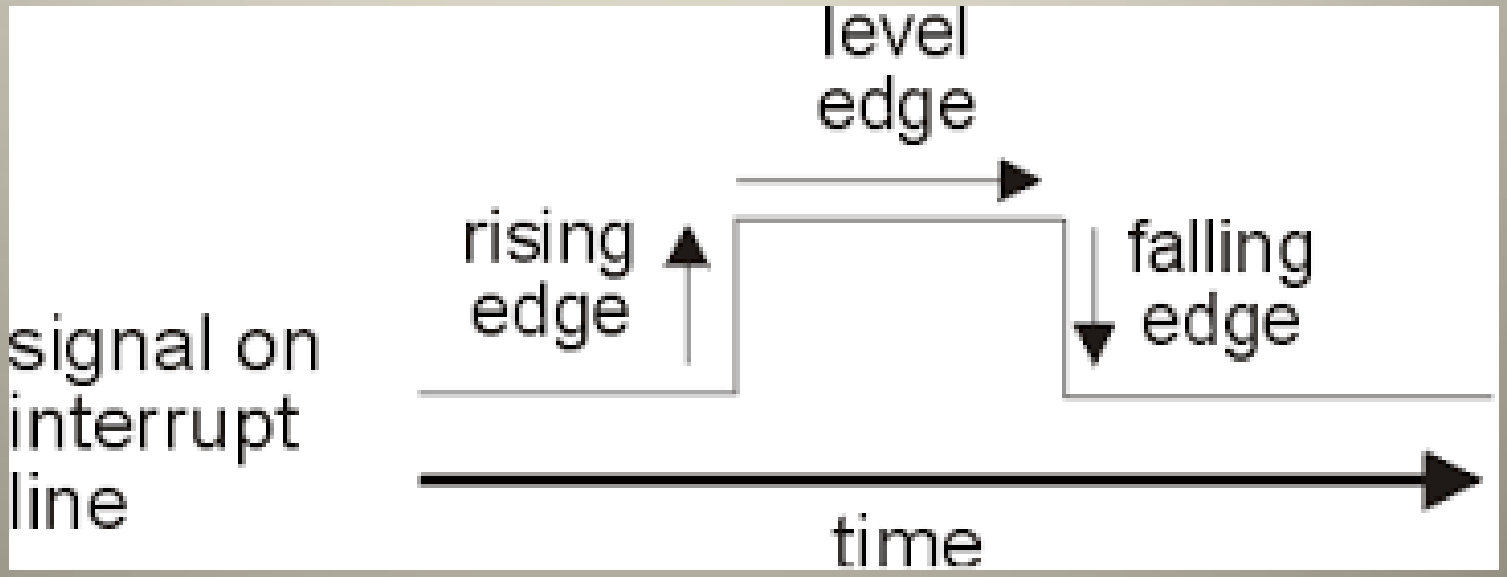

XNUCLEO-F401RE
Morpho Headers
mbed Enabled
USB TO UART RX TX
USB
DC 6-12V
LED1
LED2
LED3
LED4
PWR
RESET
USER
PC_10
PC_12
+3.3V
BOOT0
SPI3
SCK
MOSI
PA_13
PA_14
PA_15
PWM2/1
SPI1 NSS
GND
Serial1 RX
PB_7
PWM4/2
I2C1 SDA
PC_13
USER_BUTTON
PC_14
PC_15
PH_0
PH_1
VBAT
PC_2
PC_3
Analog In
SPI2
MISO
MOSI
SPI3 MISO
PC_11
PD_2
E5V
GND
NC
IOREF
NRST
+3v3
+5v
GND
GND
VIN
same as Arduino CN6 header
same as Arduino CN8 header
PA_0
PA_1
PA_4
PB_0
PC_1
PC_0
PC_9
PB_8
PB_9
AVDD
GND
PA_5
PA_6
PA_7
PB_6
PC_7
PA_9
PA_8
PB_10
PB_4
PB_5
PB_3
PA_10
PA_2
PA_3
PWM3/4
I2C3 SDA
same as Arduino CN5 header
same as Arduino CN9 header
Serial6 TX
Analog In
PWM3/3
PWM3/1
PC_8
PC_6
PC_5
U5V
Serial6
RX
TX
PWM1/4
SPI2 NSS
PA_12
PA_11
PB_12
GND
PB_2
PB_1
PB_15
PB_14
PB_13
AGND
PC_4
Analog In
PWM1/3N
PWM1/3N
PWM1/2N
PWM1/1N
SPI2
MOSI
MISO
SCK
Analog In
CN7
CN8
CN9
CN10



The hardware of controller EXTI: blue boxes are registers, red dots are input pins

Segnali interrupt

Per il micro STM32, i segnali di interrupt partono solo se il segnale cambia di stato da basso verso l'alto (rising) o dall'alto verso il basso (falling)



STM32 parametro fall

```
#include "mbed.h"
InterruptIn mybutton(USER_BUTTON);
DigitalOut myled(LED1);
float delay = 1.0; // 1 sec
void pressed()
{
   if (delay == 1.0)
      delay = 0.2; // 200 ms
   else
      delay = 1.0; // 1 sec
}
int main()
{
   mybutton.fall(&pressed);     //pressed è la function che viene attivata nel caso di interrupt
   while (1) {
      myled = !myled;
      wait(delay);
   }
}
```

STM32 parametro rise

```
#include "mbed.h"
InterruptIn mybutton(USER_BUTTON);
DigitalOut myled(LED1);
float delay = 1.0; // 1 sec
void pressed()
{
   if (delay == 1.0)
      delay = 0.2; // 200 ms
   else
      delay = 1.0; // 1 sec
}
int main()
{
   mybutton.rise(&pressed); );     //pressed è la function che viene attivata nel caso di interrupt
   while (1) {
      myled = !myled;
      wait(delay);
   }
}
```

Interrupt sul pin A5

```
#include "mbed.h"
InterruptIn button(PA_5); //define and name the interrupt input
DigitalOut led(LED1);
DigitalOut flash(LED4);
void ISR1() { //this is the response to interrupt, i.e. the ISR
   led = !led;
}
int main() {
   button.rise(&ISR1); // attach the address of the ISR function to the
   // interrupt rising edge
   while(1) { // continuous loop, ready to be interrupted
      flash = !flash;
      wait(1);
   }
}
```

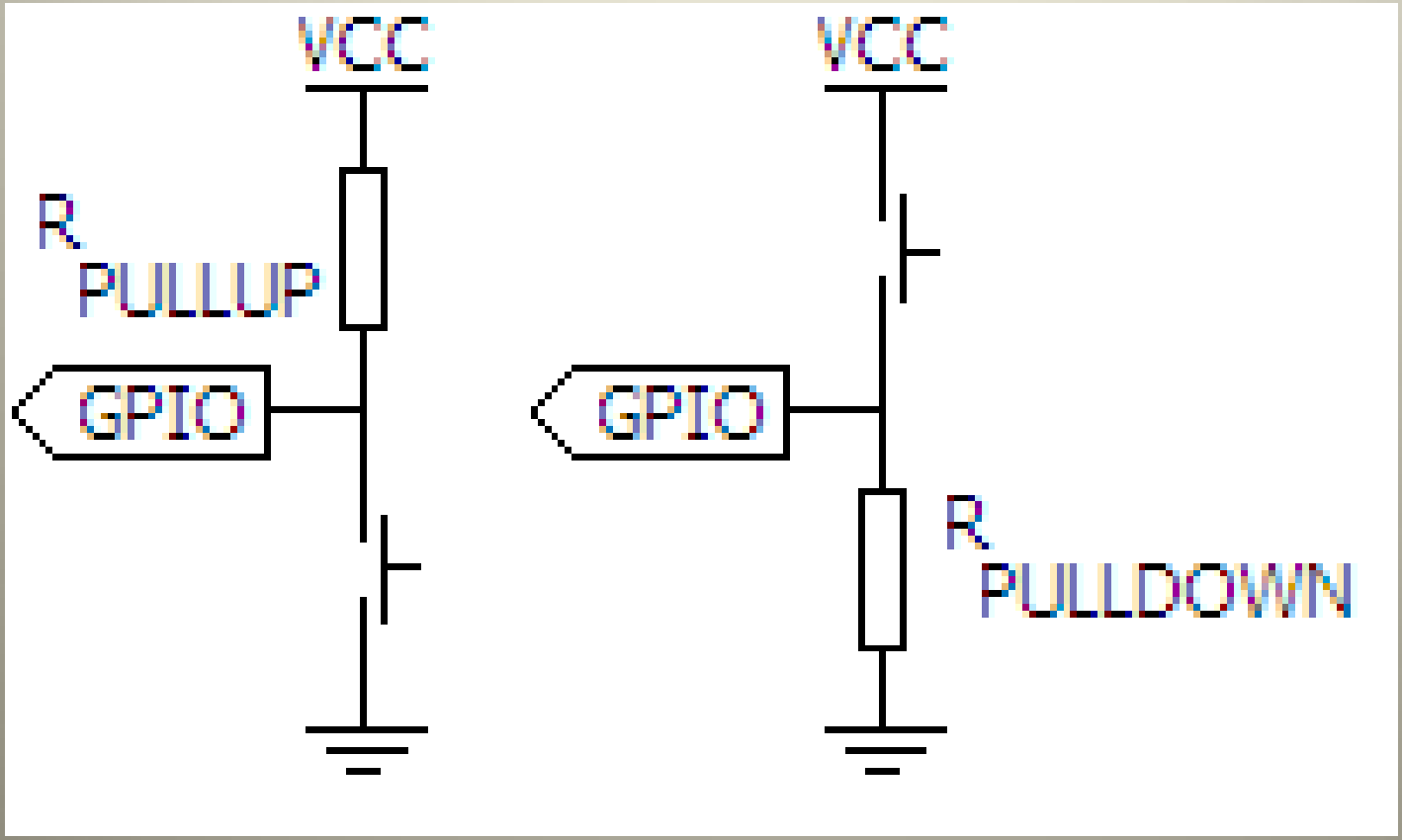
Interrupt Raspberry pi

- A differenza di Arduino e STm32, l'interrupt su Raspberry è una questione di software e non di hardware.
- Per poter gestire una operazione di interrupt bisogna attivare i resistori di PULL UP o PULL DOWN tramite software

Raspberry pi: Resistori di PULL UP e PULL DOWN

- In generale, i resistori Pull Up e Pull Down sono utilizzati nei circuiti logici e nei micro per fornire una tensione di riferimento quando si devono confrontare differenti livelli di tensione in ingresso.
- I pin di ingresso dei circuiti digitali presentano normalmente un'impedenza abbastanza alta, e ciò comporta che basta una corrente minima per poter cambiarne lo stato. Inoltre, i dispositivi collegati su di essi possono captare qualsiasi disturbo elettromagnetico o elettrostatico e quindi a commutare le loro uscite in modo del tutto imprevisto ed incontrollato e trovarsi quindi in uno stato fluttuante detto anche floating
- Se un pin di ingresso non ha una tensione di riferimento, potrebbe rilevare un segnale contrario a quello reale, dando risultati imprevedibili o addirittura dannosi per il circuito stesso.
- Raspberry Pi è dotato di resistori pull-up e pull-down attivabili via software così da poter eliminare resistori "fisici" per i pulsanti.

PULL UP e PULL DOWN



Interrupt Raspberry pi

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
# GPIO 23 pull up sul pin 23
GPIO.setup(23, GPIO.IN,
pull_up_down=GPIO.PUD_UP)
GPIO.setup(18,GPIO.OUT)
print ("GPIO sul pin 23 (pin 16) e GND (pin 6)\n")
```

Interrupt Raspberry

```
while True:
    GPIO.output(18,True)
    try:
        GPIO.wait_for_edge(23, GPIO.FALLING)
#l'interruzione avviene quando viene premuto il pulsante
        GPIO.output(18,False)
#il led viene spento per 1 secondo
        time.sleep(1)
    except KeyboardInterrupt:
        GPIO.cleanup()
GPIO.cleanup()
```

Raspberry modalità di interrupt

- GPIO.wait_for_edge(23, GPIO.FALLING)
- GPIO.wait_for_edge(23, GPIO.RISING)
- GPIO.wait_for_edge(23, GPIO.BOTH)